

## Six Sigma for Software

Richard E. Biehl

Six Sigma concepts are increasingly penetrating software engineering literature and practice. But exactly what Six Sigma is and how you apply its concepts remain unclear. To make sense of the Six Sigma movement,<sup>1</sup> we must look back to the origins and implications of its dominant predecessor, Total Quality Management.

### Brute-force quality



Historically, quality improvement was a management-dictated process of applying brute-force effort to particular quality problems. For example, management would set a goal of reducing back orders in an order-processing environment by 50 percent—from the 24 percent rate to a target of 12 percent or less. This goal would drive an all-out effort to attack the problem by making changes in the problem area and observing their impact on the back-order rate. The rate would eventually drop to the 12 percent target or below, and management would declare the improvement process a success on the basis of this result (see Figure 1). However, the improvement process's actual behavior would have varied considerably. At times there would be more back orders and at other times there would be far fewer. The process would likely remain unpredictable—causing a fluctuating back-order rate—and yet management would be happy because the overall rate would stay considerably below 24 percent for some time.

Problems with the brute-force approach are numerous but center on the fact that such efforts often focus on incorrect or inappropriate solutions that usually aren't even sustainable. You can solve virtually any quality problem in the short run by altering existing controls, special expediting, or off-cycle planning. For example, you can reduce back orders by manufacturing more of the commonly back-ordered products, running special reports to expedite problem orders, or overriding control policies that might prevent certain product substitutions. The result will be more orders out the door but potentially at the cost of a suboptimized system. These changes could lead to scheduling problems with other products, profitability loss because expediting cost is added to standard cost, and decreased customer satisfaction as ill-advised, last-minute substitutions push orders onto unwary customers. As these problems work themselves out over time through customer complaints and revised production planning, the original problem will resurface and management will find itself right back where it started with high back-order rates.

### Total Quality Management

Recognizing the brute-force approach's weaknesses caused a shift toward more systematic quality improvement approaches. These approaches consider process improvement key in satisfying the customer's perception of quality while using a fact-based approach to monitoring and decision making. These approaches became collectively known

as Total Quality Management, and many variations were popular in software quality literature from the 1980s to mid 1990s.<sup>2</sup> Quality tools that had been used for decades in other fields became popular as TQM caught on with management and penetrated product and process design in many disciplines.

TQM made a quantitative impact with Statistical Process Control,<sup>3</sup> the climactic TQM tool that brought analysis and decision making into quality improvement. SPC had two key effects:

- Quality engineers expected processes to exhibit variation close to an average value but within certain expected ranges (*control limits*).
- What customers wanted from a process (*specification limits*) wasn't necessarily what they would see the process do.

When a process is operating outside of its specification limits, it's producing *defectives*. When a process is operating outside control limits, it's *out of control*.

An out-of-control process signals a problem with the underlying process and that you should use TQM methods and tools to address the problem. So SPC analysis identifies both the location of the problems (producing defectives) and whether or not you can cost-effectively fix them (out-of-control process behaviors). Defects from in-control processes are harder to isolate and correct and require a different approach to systematically using the TQM toolset.

Figure 2a illustrates the back-order problem using basic SPC thinking. Management's 12 percent target becomes the *upper specification limit* (USL) of the desired new process. The objective will be to build a process that results in a back-order rate not exceeding this value, making the design target's *upper control limit* (UCL) also 12 percent. (You can't calculate actual control limits until the process is in place, so control limits are discussed as design targets.) Presumably, management would want to reduce the back-order rate as much as possible (*lower-*

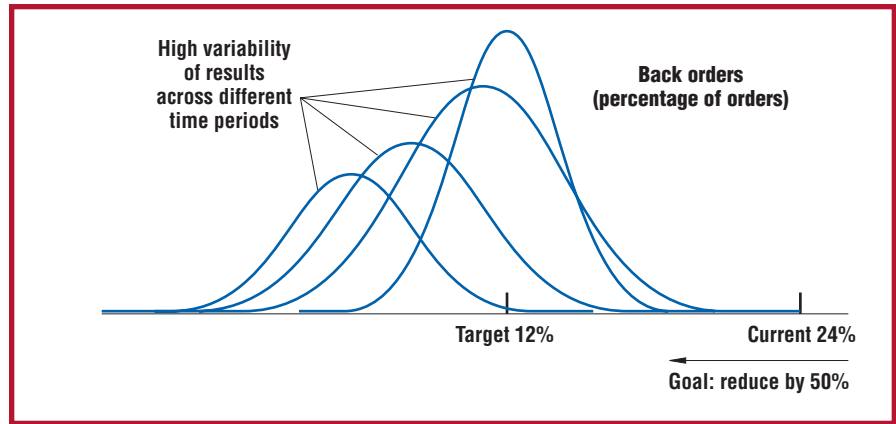


Figure 1. Goal-based metric target with unpredictable process behaviors.

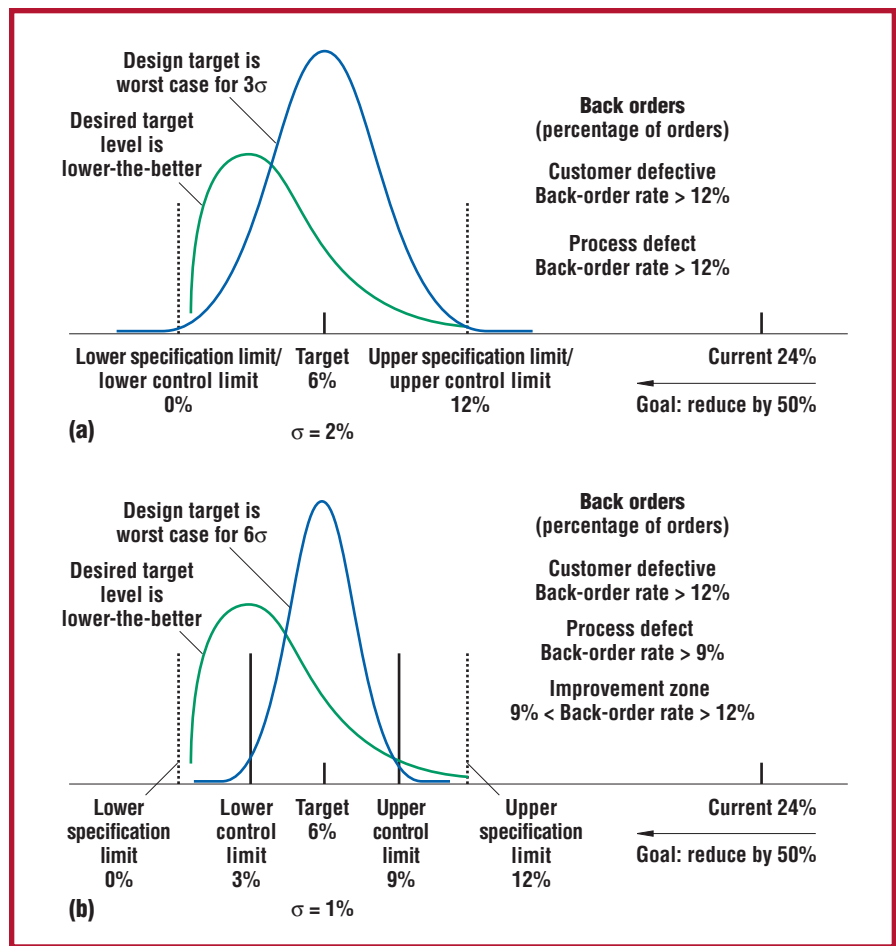


Figure 2. Back-order rate example: (a) Total Quality Management improvement metric based on three standard-deviation control limits in Statistical Process Control. (b) Six Sigma metric with improvement zone between three standard-deviation control and six standard-deviation specification limits.

*the-better*), and so the *lower specification limit* and *lower control limit* are both 0 percent. By convention in SPC, the target value for the process re-

design is the midpoint of the two specification limits, or 6 percent. The new process should deliver a 6 percent back-order rate. Fluctuations should be so

minimal that variations within three standard deviations ( $3\sigma$ ) from the mean should still be under the 12 percent USL.

So, SPC guides improvement toward a process that exhibits the natural range of variation while still producing products or services that meet customers' expectations for quality. The resulting process will exhibit a  $3\sigma$  quality level, and the SPC data will show quality improvement opportunities by monitoring defectives and process variability.

### Six Sigma

Six Sigma differs from TQM in its emphasis on raising the bar on quality. The processes designed in TQM initiatives became sensitive to  $3\sigma$  control exceptions in SPC, with ongoing improvement occurring incrementally at these margins. Six Sigma uses all the TQM tools and techniques and adds an emphasis on long-term process variability and shift.

With TQM, processes that were in control in the short run would become out of control in the long run as their variability increased with human error, equipment wear, and gradual deterioration of process conditions. Software variability might result as user skills didn't keep up with changing software features, response times degraded because of increasing network loads, or databases became less efficient as the relative volume of historical versus active data changed.


With increased variability, TQM models failed to deliver adequate quality, even at short-term  $3\sigma$  levels. The short-term expected defect rate of less than 1 percent for  $3\sigma$  processes could rise above 5 percent because of long-term process shifts. By broadening expectations to  $6\sigma$  quality, new processes could provide acceptable quality levels while accommodating the effects of long-term process shift.

Quality engineers still use SPC to monitor and evaluate process performance at  $3\sigma$  levels. However, the identified exceptions are now occurring well within the  $6\sigma$  specification limits. In TQM, process defects and customer

defectives were both defined at  $3\sigma$ , which necessitated process improvement while dealing with customer defectives outside the process. To let processes and systems self-correct and adjust to results in the  $3\sigma$ -to- $6\sigma$  range, Six Sigma separates discussing process defects (outside  $3\sigma$ ) from recognizing customer defectives (outside  $6\sigma$ ). Such self-correcting processes actively measure their own performance and have additional reaction procedures for when key metrics fall in the defined improvement zone. For example, a data display application might temporarily reduce the amount of detail it displays per screen if it notes that data access or response times are rising above their control limits.

Figure 2b illustrates this difference using the back-order rate example. The specification limits haven't changed because they represent the customer's desires, which don't depend on how you measure quality. But the control limits change. Design target SPC control limits are still  $3\sigma$  from the target, although the specification limits are now  $12\sigma$  apart in this new Six Sigma view. This means that the revised UCL is now 9 percent, or the midpoint between the 6 percent target value and 12 percent USL. An improvement zone now exists between the UCL and USL. Values above the control limit are process defects that SPC says can be economically

corrected. If you can correct them before they exceed the USL, the customer need never see a defective. As I mentioned earlier, control limits are design targets and you can't measure actual performance until you implement the design. Cutting a design target control limit in half isn't trivial; it entails shifting the level of customer-perceived performance from 60 to 70 deviations-per-thousand to 3 to 4 deviations-per-million. Roughly half of the acceptable performance observations under the  $3\sigma$  design become process defects under the  $6\sigma$  design. Organizations that can achieve such tight performance in key design dimensions can yield enormous benefits.

As software engineers redesign processes in line with Six Sigma, they can implement controls that take advantage of the improvement zone between  $3\sigma$  and  $6\sigma$  process performance. By building critical customer metrics into software solutions (for example, response times, cycle times, transaction rates, access frequencies, and user-defined thresholds), they can make applications self-correcting by enabling specific actions when process defects surface in the improvement zone. These actions don't always need sophisticated technical solutions to be beneficial. Controls can be as simple as an email notifying support personnel of defects above the  $3\sigma$  level or a periodic report-highlighting activity in the  $3\sigma$ -to- $6\sigma$  zone. The point isn't to build software without defects but to prevent software from producing defectives in spite of their defects. That's the essence of Six Sigma for software. 

**The point isn't to build software without defects but to prevent software from producing defectives in spite of their defects.**

### References

1. C.B. Tayntor, *Six Sigma Software Development*, Auerbach Publications, 2002.
2. J.M. Juran, *A History of Managing for Quality*, ASQC Quality Press, 1995.
3. W.A. Florac and A.D. Carleton, "Using Statistical Process Control to Measure Software Processes," *Fundamental Concepts for the Software Quality Engineer*, T. Daughtrey, ed., ASQC Quality Press, 2003, pp. 133-143.

**Richard E. Biehl** is a quality advisor at Data-Oriented Quality Solutions. Contact him at [rbiehl@doqs.com](mailto:rbiehl@doqs.com).